

8 ПРОГРАМУВАННЯ ПІДПРОГРАМ

Мета: навчитись розробляти і використовувати на практиці власні функції для вирішення завдань, які стоять перед розробником програм.

8.1 Короткі теоретичні відомості

Поняття функції та її складові. *Функція* – поіменована послідовність описів і операторів, яка виконує деяку закінчену послідовність дій. Будь-яка функція складається із заголовка (оголошення функції) і тіла (визначення функції). Оголошення функції описує її прототип (іноді кажуть "сигнатура"). Прототип функції оголошується наступним чином:

Тип_Повернення Ім'я_Функції (Список Оголошених Параметрів);

Тут Тип Повернення – тип даних, що повертається функцією. Якщо він не зазначений, то за замовчуванням вважається, що повертається тип `int`.

Список Оголошених Параметрів - задає тип і ім'я кожного з параметрів функції, розділених комами (ім'я параметра можна опускати). Список параметрів функції може бути порожнім.

Приклади прототипів функцій: `double max (double par1, double par2); int swap(int, int) ; void func() ;`

Визначення функції складається з її заголовка і власне тіла, вкладеного у фігурні дужки і такого, що має смислове навантаження. Якщо функція повертає будь-яке значення, в тілі функції обов'язково повинен бути присутнім оператор повернення з параметром того ж типу.

Тип_Повернення Ім'я_Функції (Список Оголошених Параметрів)

{ // тіло функції }

Виклик функції – вказівка ідентифікатора функції (її імені), за яким в круглих дужках слідує список аргументів, розділених комами. `double maxValue = max (2.5, 1235.2); int j = swap(10, 2); func();`

Функції, що не повертають значення. Це функції типу `void` – ті, що не повертають значення – можуть розглядатися, як деякий різновид команд, реалізований особливими програмними операторами.

Оператор `func();` виконує функцію `void func()` , тобто передасть керування функції, доки не виконаються усі її оператори. Коли функція поверне керування в основну програму, тобто завершить свою роботу, програма продовжить своє виконання з того місця, де розташовується наступний оператор за оператором `func()`. Якщо функція повертає значення типу `void`, то її виклик слід організовувати так, щоб значення, яке повертається, не використовувалося. Тобто, таку функцію не використовують у правій частині виразу.

Аргументи функції за замовчуванням. С++ допускає при виклику функцій опускати деякі її параметри. Досягається це зазначенням в прототипі функції значень аргументів за замовчуванням. Наприклад, функція, прототип якої наведено нижче, може при виклику мати різний вигляд в залежності від ситуації.

```
// Прототип функції:
void ShowInt(int i,bool Flag=true,char symbol='\n');
// Виклик функції ShowInt:
ShowInt(A, false, 'a');
ShowInt(B, false);
ShowInt(C);
```

Передавання параметрів у функцію. Механізм передавання параметрів є основним способом обміну інформацією між функціями. Параметри, перераховані в заголовку опису функції, називаються формальними параметрами (або просто параметрами), а записані в операторі виклику функції – фактичними параметрами (або аргументами).

Існує два способи передачі параметрів у функцію: за значенням і за адресою. При передаванні за значенням в стек заносяться копії значень аргументів, і оператори функції працюють з цими копіями. Доступу до початкових значень параметрів у функції немає, а, отже, немає і можливості їх змінити. При передаванні за адресою в стек заносяться копії адрес аргументів, а функція здійснює доступ до комірок пам'яті за цими адресами і може змінити значення аргументів:

```
#include <iostream>
void f(int i, int* j, int& k);
int main()
{
    int i=1,j=2,k=3;
    cout<<i<<' '<<j<<' '<<k<<endl; // Результат: 1 2 3
    f(i,&j,k);
    cout<<i<<' '<<j<<' '<<k<<endl; // Результат: 1 3 4
    return 0;
}
void f(int i, int* j, int& k) { i++; (*j)++; k++; }
```

Перший параметр (i) передається за значенням. Його зміна в функції не

впливає на вихідне значення. Другий параметр (j) передається за адресою за допомогою покажчика, при цьому для передавання у функцію адреси фактичного параметра використовується операція взяття адреси, а для отримання його значення в функції потрібна операція розіменування. Третій параметр (k) передається адресою за допомогою посилання.

При передаванні за посиланням у функцію передається адреса зазначеного при виклику параметра, а всередині функції всі звернення до параметру неявно розіменовуються. Використання посилань замість покажчиків, по-перше, покращує читабельність програми, позбавляючи від необхідності застосовувати операції одержання адреси та розіменування; і, по-друге, не вимагає копіювання параметрів, що важливо при передаванні структур даних великого обсягу. Якщо потрібно заборонити зміну параметра усередині функції, використовується модифікатор `const`:

```
int f(const char*);  
char* t(char* a, const int* b);
```

Масиви як параметри функції, за винятком параметрів типу покажчик та масив, передаються за значенням. Це означає, що під час виклику функції їй передаються тільки значення змінних. Сама функція не в змозі змінити цих значень у функції, що їх викликає. Одновимірний масив як параметр Ім'я масиву є покажчиком на його нульовий елемент, тому у функцію масиви передаються за покажчиком. Кількість елементів масиву може передаватися окремим параметром.

8.2 Виконання завдання

Завдання 1. Програма перевіряє, чи ортогональні вектори $x = \{x_n\}$ і $y = \{y_n\}$.

Лістинг 12.1 – Вихідний код завдання 1

```
#include <iostream>  
using namespace std;  
  
void getRandomVector(int*Arr, int n) {  
    for (int i = 0; i < n; i++) Arr[i] = rand() % 10;  
}  
void showVector(int * Arr, int n) {  
    for(int i=0;i<n;i++)cout<<Arr[i]<<" ";  
    cout << endl;  
}  
  
bool checkOrthogonality (int * arrx, int * array, int n){  
    int scalar = 0;  
    for (int i = 0; i < n; i++) scalar += arrx[i] * array[i];  
    if (scalar == 0) return true; else return false;  
}
```

```

int main() {
    int n;
    cout << "Enter array lenght: n = ";
    cin >> n;
    int* x = new int[n];
    int* y = new int[n];

    getRandomVector(x, n); getRandomVector(y, n);
    showVector(x, n); showVector(y, n);
    if (checkOrthogonality(x, y, n)) {
        cout << "Arrays are perpendicular." << endl;
    } else {
        cout << "Arrays are't perpendicular." << endl;
    }

    delete[]x; delete[]y;
    system("pause");
    return 0;
}

```

У ході виконання даної програми отримано наступні результати:

```

Enter array lenght: n = 5
7 9 3 8 0
2 4 8 3 9
Arrays are't perpendicular.
sh: pause: command not found
Program ended with exit code: 0

```

Рисунок 8.1 — Результат обчислень

Завдання 2. Створення нової матриці, кожний елемент якої дорівнює відповідному елементу матриці $A_{n \times n}$, розділеному на найбільший елемент цієї матриці.

```

#include <iostream>
using namespace std;

void getRandomMatrix(int**Arr, int n) {
    for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++) Arr[i][j] = rand() % 101;
}

void showMatrix(int ** Arr, int n) {
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) cout << Arr[i][j] << '\t';
        cout << endl;
    }
}

```

```

int getLargestElement(int ** Arr, int n) {
    int max, tmp;
    for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++) {
            tmp = Arr[i][j];
            if (max < tmp) max = tmp;
        }
    return max;
}

void devideByLargestElement(int ** ArrA, int ** ArrB, int n, int max) {
    for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++) ArrB[i][j] = ArrA[i][j] / max;
}

int main() {
    int n = 5;
    int** A = new int*[n];
    int** B = new int*[n];
    for (int i=0;i<n;i++) B[i] = new int[n];
    for (int i=0;i<n;i++) A[i] = new int[n];

    getRandomMatrix(A, n);
    cout << "A: " << endl; showMatrix(A, n);
    devideByLargestElement(A, B, n, getLargestElement(A, n));
    cout << "B: " << endl; showMatrix(B, n);

    for (int i = 0; i < n; i++) {delete A[i]; delete B[i];}
    delete[]A; delete[]B;
    system("pause");
    return 0;
}

```

У ході виконання даної програми отримано наступні результати:

```

A:
41  65  31  41  19
15  72  11  78  69
37  23  29  63  75
4   5   49  75  99
27  61  62  17  79
B:
0   0   0   0   0
0   0   0   0   0
0   0   0   0   0
0   0   0   0   1
0   0   0   0   0
sh: pause: command not found
Program ended with exit code: 0

```

Рисунок 8.2 — Результат обчислень

8.3 Програми та обладнання.

Xcode, OpenOffice

8.4 Висновки.

Під час виконання даної практичної роботи були здобуті навички розроблення і використання на практиці власних функцій для вирішення завдань, які стоять перед розробником програм.